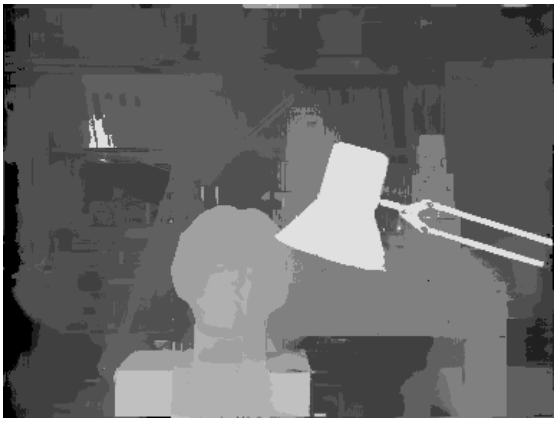
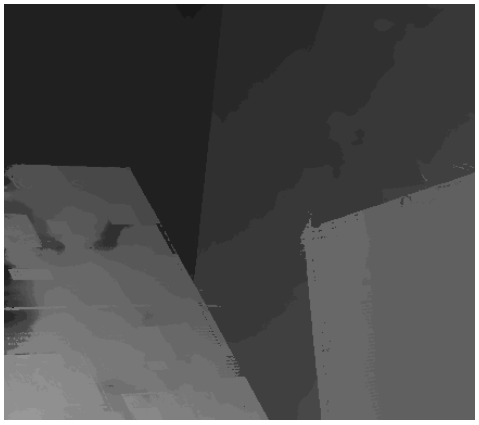
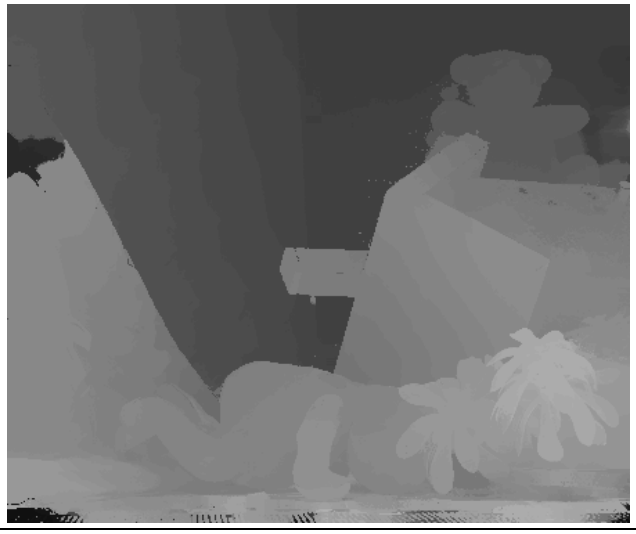
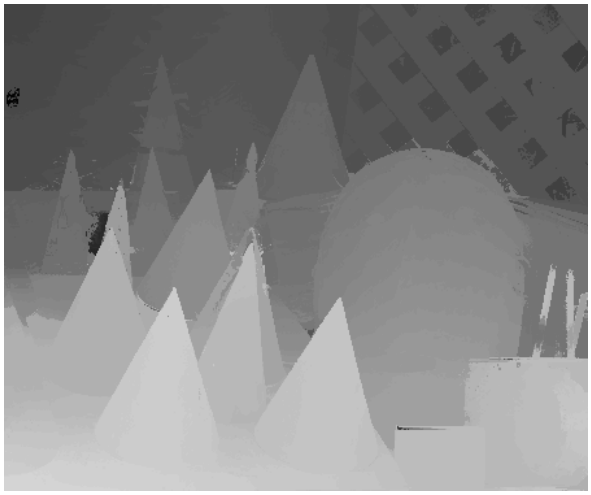
Computer Vision HW4 Report

Student ID: B08505024
Name: 劉虹伶

Visualize the disparity map of 4 testing images.

Tsukuba(2min20sec)	Venus(4min22sec)
	
Teddy(5min8sec)	Cones(7min9sec)
	

Report the bad pixel ratio of 2 testing images with given ground truth (Tsukuba/Teddy).

	bad pixel ratio
Tsukuba	3.49%
Teddy	11.20%

Describe your algorithm in terms of 4-step pipeline.

- **Cost computation**

  先 zero_padding，補齊 Census cost 最旁邊的 pixel，接者用兩圈 loop 去計算每個 pixel 的 Local binary pattern(call function : computeLBP)，如果有 channel 的話會分開計算。

  接著用 range(max_disp+1)的 loop 計算 Hamming distance：

  1、LBP_IL!=LBP_IR 可以把不同的地方變成 1

  2、np.count_nonzero 直接算出變出有多少個 1（多 channel 的話也會直接合併計算）

```
# >>> Cost Computation
Create Jira Issue
# TODO: Compute matching cost
# [Tips] Census cost = Local binary pattern -> Hamming distance
# [Tips] Set costs of out-of-bound pixels = cost of closest valid pixel
# [Tips] Compute cost both "Il to Ir" and "Ir to Il" for later left-right consistency
 # >>> Cost Aggregation
Create Jira Issue
# TODO: Refine the cost according to nearby costs
# [Tips] Joint bilateral filter (for the cost of each disparty)
Il_pad = cv2.copyMakeBorder(Il, 1, 1, 1, 1, cv2.BORDER_CONSTANT, (0,0,0))
Ir_pad = cv2.copyMakeBorder(Ir, 1, 1, 1, 1, cv2.BORDER_CONSTANT, (0,0,0))
LBP_IL=np.zeros((h,w,ch,8))
LBP_IR=np.zeros((h,w,ch,8))

for x in (range((w))):
    for y in range(h):
        LBP_IL[y][x]=computeLBP(img=Il_pad,ch=ch,x=x+1,y=y+1)
        LBP_IR[y][x]=computeLBP(img=Ir_pad,ch=ch,x=x+1,y=y+1)

cost_map_L2R=np.ones(((h,w,max_disp+1)), dtype=np.float32)*24
cost_map_R2L=np.ones(((h,w,max_disp+1)), dtype=np.float32) *24

for disp in (range(max_disp+1)):
        if(disp==0):
            cost_map_L2R[...,disp] = np.count_nonzero(LBP_IL!=LBP_IR,axis=(2,3))
            cost_map_R2L[...,disp] = np.count_nonzero(LBP_IR!=LBP_IL,axis=(2,3))
        else:
            cost_map_L2R[:,disp:,disp] = np.count_nonzero(LBP_IL[:,disp:,:,:]!=LBP_IR[:,:-disp,:,:],axis=(2,3))
            cost_map_R2L[:,:-disp,disp] = np.count_nonzero(LBP_IR[:,:-disp,:,:]!=LBP_IL[:,disp:,:,:],axis=(2,3))
        cost_map_L2R[...,disp]=xip.jointBilateralFilter(Il,cost_map_L2R[...,disp],10,10,10)
        cost_map_R2L[...,disp]=xip.jointBilateralFilter(Ir,cost_map_R2L[...,disp],10,10,10)
```

```
def computeLBP(img,ch,x,y):
    img=np.array(img)
    LBP_arr=[]
    LBP=[]

    center=img[y][x]

    #shape =(8,ch)
    LBP_arr.append(img[y+1][x-1])     # top_right
    LBP_arr.append(img[y+1][x])       # right
    LBP_arr.append(img[y+1][x+1])     # bottom_right
    LBP_arr.append(img[y][x+1])       # bottom
    LBP_arr.append(img[y-1][x+1])     # bottom_left
    LBP_arr.append(img[y-1][x])       # left
    LBP_arr.append(img[y-1][x-1])     # top_left
    LBP_arr.append(img[y][x-1])       # top

    LBP_arr=np.array(LBP_arr)

    LBP_slice=[]
    for i in range(ch):
        LBP_slice=LBP_arr[...,i]
        LBP_slice[LBP_slice>center[i]]=1
        LBP_slice[LBP_slice!=1]=0
        LBP.append(LBP_slice)

    LBP=np.array(LBP)

    return LBP
```

- **Cost aggregation**

直接寫在 range(max_disp+1)的 loop 最後面，這樣不用多跑一次 loop，jointBilateralFilter 的參數會改變結果(如下圖)

parameters	Tsukuba	Teddy
(10,10,10)	**3.49%**	**11.20%**
(15,10,10)	3.83%	11.15%
(30,10,10)	4.84%	9.99%

```
# >>> Cost Computation
Create Jira Issue
# TODO: Compute matching cost
# [Tips] Census cost = Local binary pattern -> Hamming distance
# [Tips] Set costs of out-of-bound pixels = cost of closest valid pixel
# [Tips] Compute cost both "Il to Ir" and "Ir to Il" for later left-right consistency
# >>> Cost Aggregation
Create Jira Issue
# TODO: Refine the cost according to nearby costs
# [Tips] Joint bilateral filter (for the cost of each disparity)
Il_pad = cv2.copyMakeBorder(Il, 1, 1, 1, 1, cv2.BORDER_CONSTANT, (0,0,0))
Ir_pad = cv2.copyMakeBorder(Ir, 1, 1, 1, 1, cv2.BORDER_CONSTANT, (0,0,0))
LBP_IL=np.zeros((h,w,ch,8))
LBP_IR=np.zeros((h,w,ch,8))

for x in (range((w))):
    for y in range(h):
        LBP_IL[y][x]=computeLBP(img=Il_pad,ch=ch,x=x+1,y=y+1)
        LBP_IR[y][x]=computeLBP(img=Ir_pad,ch=ch,x=x+1,y=y+1)

cost_map_L2R=np.ones(((h,w,max_disp+1)), dtype=np.float32)*24
cost_map_R2L=np.ones(((h,w,max_disp+1)), dtype=np.float32) *24

for disp in (range(max_disp+1)):
        if(disp==0):
            cost_map_L2R[...,disp] = np.count_nonzero(LBP_IL!=LBP_IR,axis=(2,3))
            cost_map_R2L[...,disp] = np.count_nonzero(LBP_IR!=LBP_IL,axis=(2,3))
        else:
            cost_map_L2R[:,disp:,disp] = np.count_nonzero(LBP_IL[:,disp:,:,:]!=LBP_IR[:,:-disp,:,:],axis=(2,3))
            cost_map_R2L[:,:-disp,disp] = np.count_nonzero(LBP_IR[:,:-disp,:,:]!=LBP_IL[:,disp:,:,:],axis=(2,3))
        cost_map_L2R[...,disp]=xip.jointBilateralFilter(Il,cost_map_L2R[...,disp],10,10,10)
        cost_map_R2L[...,disp]=xip.jointBilateralFilter(Ir,cost_map_R2L[...,disp],10,10,10)
```

• Disparity optimization

用 np.argmin 找 cost 最小的

```
# >>> Disparity Optimization
Create Jira Issue
# TODO: Determine disparity based on estimated cost.
# [Tips] Winner-take-all
Winner_cost_L2R=np.argmin(cost_map_L2R,axis=2)
Winner_cost_R2L=np.argmin(cost_map_R2L,axis=2)
```

• Disparity refinement

Left-right consistency check，不合的 cost 改成-1，接著直接 Hole filling(沒有很多 hole，計算量不大)，最後再過 weightedMedianFilter，步驟和 tip 一樣。

```
# >>> Disparity Refinement
Create Jira Issue
# TODO: Do whatever to enhance the disparity map
# [Tips] Left-right consistency check -> Hole filling -> Weighted median filtering
for x in range(w):
    for y in range(h):
        if (Winner_cost_L2R[y,x] != Winner_cost_R2L[y,x-Winner_cost_L2R[y,x]]):
            Winner_cost_L2R[y,x]=-1

for x in range(w):
    for y in range(h):
        FR=max_disp
        FL=max_disp
        if (Winner_cost_L2R[y,x]==-1):
            #left
            for left in range(max_disp+1):
                if( x-left < 0 ):break
                elif( Winner_cost_L2R[y,x-left]==-1):continue
                else:
                    FL=Winner_cost_L2R[y,x-left]
                    break
            #right
            for right in range(max_disp+1):
                if( x+right == w ):break
                elif( Winner_cost_L2R[y,x+right]==-1):continue
                else:
                    FR=Winner_cost_L2R[y,x+right]
                    break
            Winner_cost_L2R[y,x] = min(FL, FR)

labels = xip.weightedMedianFilter(Il.astype(np.uint8), Winner_cost_L2R.astype(np.uint8), 18, 1)
```